DOM - document object model - when we are open any web page, and open web tool all the information that we see is a

DOM. DOM is generated at runtime. Whenever you request the web page, the browser generate it, and the server sends the source code for it.

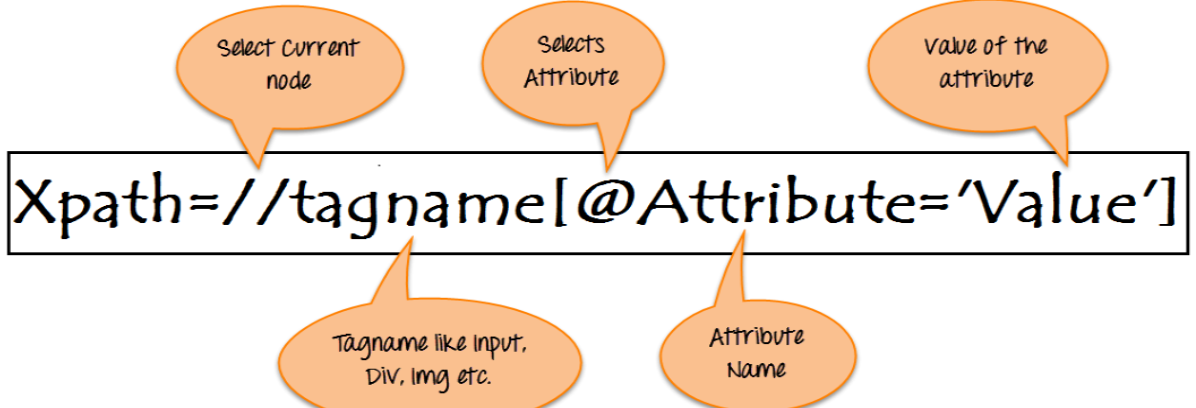
X-path works based on the DOM. Every element in the DOM has an address, and that address we need to specify to find the element.

**Xpath is a kind of navigation from the root element of an HTML page through multiple tags (elements) to the element that we are looking for.**

node - one more name for the tag, so you may say: tag, element, node it's all the same. Without tag it is not possible to place any web element to the web page.

Root element - starting point of the page!

All the elements are located in the body of the HTML document. Every tag in the DOM represent element.



XPath contains the path of the element situated at the web page. Standard XPath syntax for creating XPath starts with // or /

There are two types of XPath:

1) Absolute XPath //

2) Relative XPath /

Абсолютный - It is the direct way to find the element, but the disadvantage of the absolute XPath is that if there are any changes made in the path of the element then that XPath gets failed. это путь от корня документа. Первым символом в нём должен стоять “/”.

**/html/body/div/ul/li[1]**

**синтаксис**

**Relative XPath** - Относительный - starts from the middle of HTML DOM structure. **It starts with double forward slash** (//). It can search elements anywhere on the web page, means no need to write a long xpath and you can start from the middle of the HTML DOM structure. Relative Xpath is always preferred as it is not a complete path from the root element. это путь от одного элемента (не обязательно от корневого) к другому, но этот элемент как мне кажется должен либо быть проиндексирован [...], либо быть уникальным

**//main/div[1]/div[1]/div[1]/div[1]/div[1]/div[1]/div[1]/a[1]/img[1]**-Здесь main типа уникальный поэтому и работает.

**//main//a[1]/img[1]**-тоже сработало,символ '//'означает что учтены все родительские элементы данного тега

это называется типа рекурсивными спусками, но обратно-же первый тег(здесь-main)должен быть уникальным.

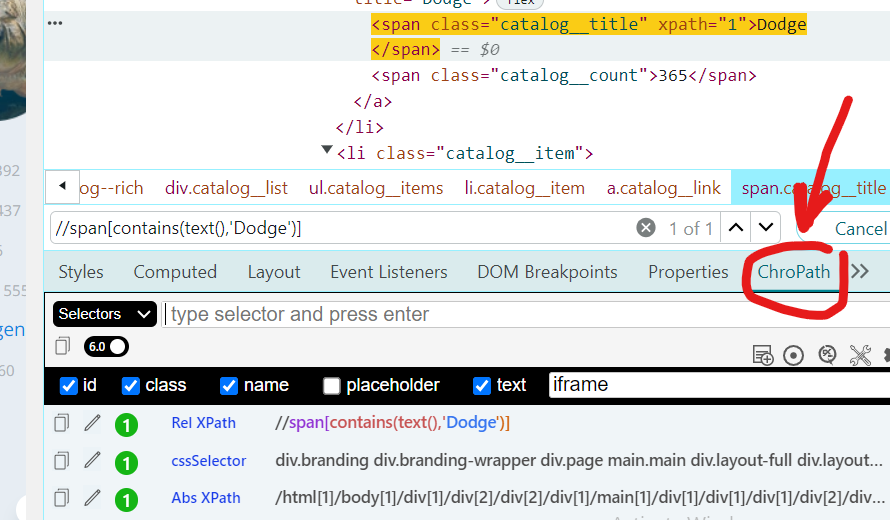
**//main//a**-тоже работает т.к. а только одна в main

**//\*//a** - тоже работает когда не знаешь название корневого элемента можно поставить \*.

**@ - значёк at the rate мы ставим когда ищем элемент по атрибуту.**

Строка XPath — это фактически путь к элементу в дереве, где каждый уровень разделяется 1косой чертой «/».

Для поиска xpath можно воспользоваться **сторонними инструментами** например ChroPath



Это расширение для браузера, его надо скачивать. Если его не видно кликни на стрелочку.

Ещё есть селектор хаб, тоже самое

**поиск дочернего элемента**

**//название\_элемента[@его\_тег=’значение’]/название\_дочернего\_элемента/…/искомый\_элемент** ←идёш от более явного элемента к вложенному через слэш

**Поиск поочерёдного элемента**

Поскольку результатом выполнения запроса XPath бывает множество элементов, то результат называется

коллекцией.//div данный XPath вернет 8 элементов (<div>). Для того, чтобы можно было работать с любым

элементом из коллекции нужно поместить в квадратные скобки номер элемента, с которым мы хотим работать.

//div[2] данный XPath вернет 2-й элемент (<div>) из коллекции элементов.Данные, заключенные в квадратные

скобки называются предикатами.

Предикаты — это фильтры. После названия узла в квадратных скобках можно указать условие фильтрации.

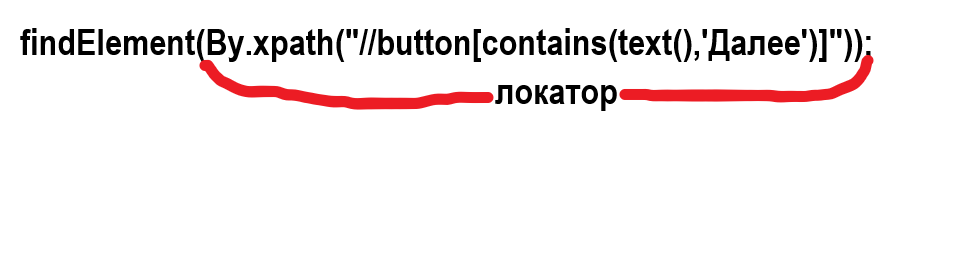
Попадают те узлы, которые соответствуют условию фильтрации

Пример://body/div[@id='\_\_next']/div[2]/div[3]

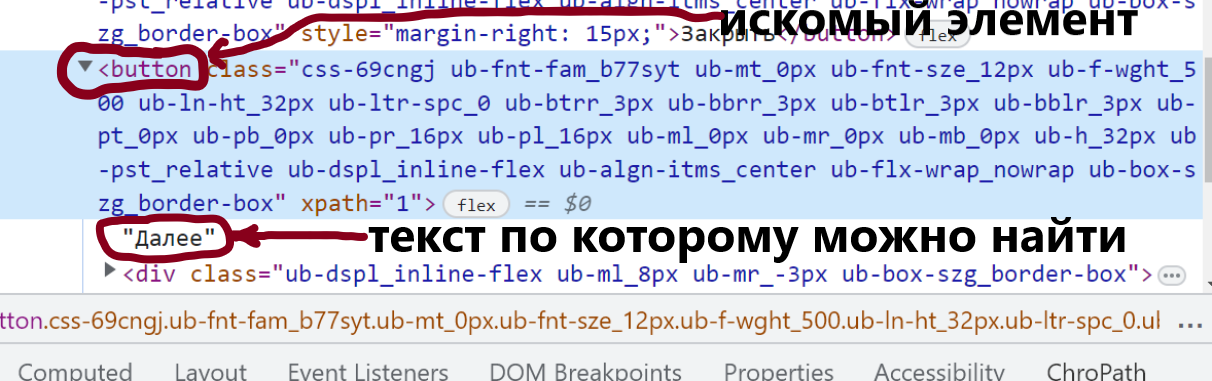
**Есть момент!**

Например: **//span[@class='catalog\_\_title']** ← этот путь возвращает 30 элементов и если я захочу выбрать к примеру 5-ый то у меня не получится сделать вот так **//span[@class='catalog\_\_title'][5]**

**Нахождение элемента по static inner тексту**

Текст который написан между открывающим тегом и закрывающим называется static inner text

Когда в элементе не за что зацепится (нету атрибутов) можно указать static inner текст



Итак оказывается есть ещо один способ нахождения по inner тексту с совсем другим синтаксисом:

**WebElement element= driver.findElement(By.xpath("//button[contains(text(),'Далее')]"));** <-- способ №1

**WebElement element=driver.findElement(By.xpath("(//button[text()='Далее'])"));**  ←способ № 2

также оказывается есть x-path function

**//a[normalize-space()=’Какой То текст’]**  ←способ № 3

также смотри икс пасс функции ниже.

\* -используется для ссылки на все элементы, которые являются дочерними для узла контекста.

Например: //\*/p ищет элемент (<p>) во всех элементах которые есть на странице.

В примере выше //div/p мы искали p только в элементе (<div>).

—--------------------—----------------------------------**поиск по цепочке**—---------------------------------------------

//nazvavie\_tega[@atribyt=’znachenie’]//nazvavie\_tega[@atribyt=’znachenie’]

первый элемент указал и к примеру последний, это благодаря **//**

**x-path functions**

**//a[normalize-space()=’Какойто текст’]** ← поиск элемента по тексту

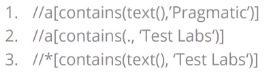
**CONTAINS** function

Принимает 2 параметра: contains( 1 parametr, 2 parametr)

1 - ый параметр это может быть или метод(text()) или атрибут

2 - ой параметр это значение

Когда мы ищем элемент по иннер тексту то он может менятся, иногда частично. есть функция позволяющая искать по части текста (менятся текст может не весь, какоето слово может быть не изменно), этот текст по которому мы ищем может быть где угодно в inner tekst (с начала или в конце или в середине) для этого есть функции:

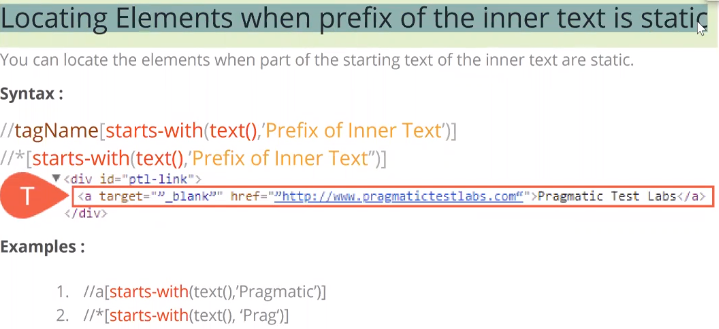


комбинация contains()+text()

**//tagname[contains (@Attribute, 'Value')]** также можно использовать при нахождении **по атрибуту**, при этом в value можно записывать частичное значение!!! Например:

**//\*[contains(@id, ‘ст’)] ←** находим элемент с атрибутом id значение которого содержит ‘ст’, при этом ст может встречаться и в начале и в конце значения. In other, words we don’t have to enter the entire statement we’re looking for; instead, we can write a few unique words on the web page.

**STARTS-WITH** function



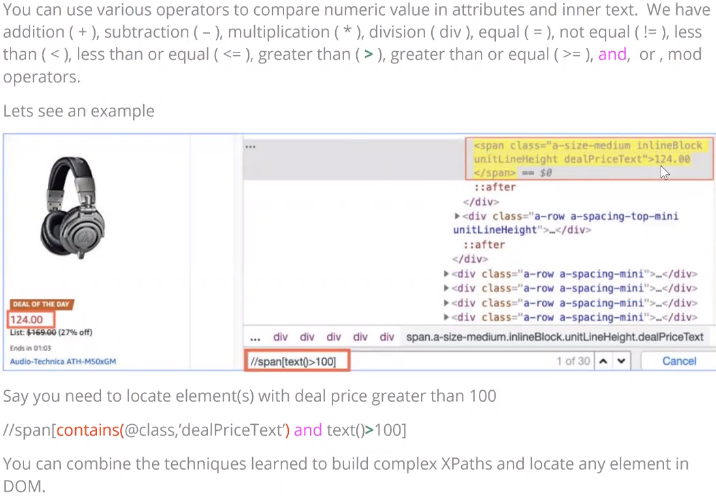
prefix - это то с чего начинается текст, хотя не,не так, это то с чего начинается первое слово иннер текста! Мы не можем взять и указать просто часть иннер текста, это можно сделать в contains(), здесь акцент именно на первое слово в тексте и даже на первую его половину.

**X-path operators**

**and, or**

Если используешь оператор то он должен стоять в одних квадратных скобках

**//\*[@id = ‘старт’ or**  **@id=’стоп’]**



**Поиск по атрибуту**

**@ - используется для ссылки на любые атрибуты.**

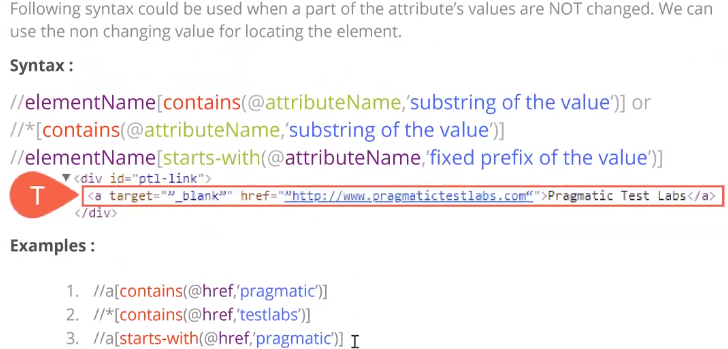
Например: **//div[@class='mainWrap']** ищет все элементы (<div>), в которых есть атрибут class

со значением mainWrap. А в css такого атрибута не требуется, там просто div[class='mainWrap']

@\* -используется для ссылки на все атрибуты узла контекста.

Например: //div[@\*] ищет все элементы div в которых есть любые аттрибуты (id, name, style...)

**поиск элементов с динамически изменяющимися значениями атрибутов**

****

Или ещё например у нас есть 1 кнопка на каторой написано старт, при её нажатии что-то происходит неважно что но текст на ней сразу меняется на стоп и если на неё нажать, что-то происходит неважно что но текст на ней опять меняется на старт. В DOM это реализовано одним элементом с тегом id, значение которого динамически меняется, когда кнопка выключена id имеет значение старт, а когда кнопка включена id = стоп. Икс пасс пишется с применением оператора or.

**//\*[@id = ‘старт’ or**  **@id=’стоп’]**

или можно использовать метод contains()

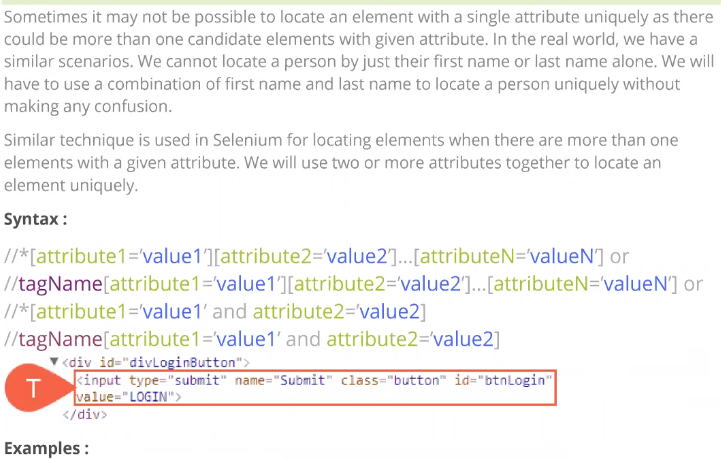
**//\*[contains(@id, ‘ст’)]**

или комбинировать эти два метода

**//\*[contains(@id, ‘ст’ or @ id, ‘st’)] ←**не уверен насчет синтаксиса но логика такая есть.

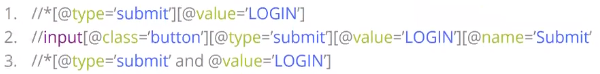
/**/nazvanie\_tega[contains(@nazvanie\_atribyta, ’znachenie’)][contains(@nazvanie\_atribyta, ‘znachenie’)]**

**Поиск элемента с несколькими атрибутами**



← Здесь не хватает **@** перед каждым атрибутом

смотри правильный синтаксис в Examples:



**← правильно**

**//a[@class=’awdsas’][@id=’dasd x’]**

**//a[@class=’awdsas’ and @id=’dasd x’]** можно использовать оператор and тогда скобки не нужны

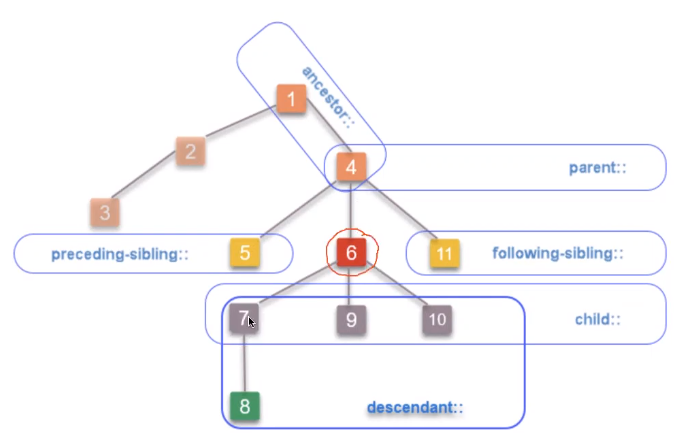
**//a[@class=’awdsas’ or @id=’dasd x’]** если хоть один атрибут = тру то икс пасс найдёт элемент

**X-path axes**

Терминология:

Это поиск элемента с помощью других, соседних, элементов (например когда нет атрибутов для зацепки)

Далее пример - нам надо найти какой то элемент без атрибутов № 6. каждая цифра это элемент. Мы можем рассматривать другие элементы в DOM вокруг 6 как ориентир, для этого надо создать терминологию какой элемент как называется по отношению к 6



6 - это **self node**, and we are identifying all other elements from 6 node.

4 - **parent tag**. (Родитель) т.е. предполагается, что 6 вложен в 4 (parent - это верхнеуровневый (на одну ступень) таг в котором находится непосредственно self node). Может быть несколько parent-ов и несколько ancestor-ов. Также 4 является parent для 5 и 11. Т.е. 5, 6, 11 вложены в 4

1 - . Это всё по отношению к 6 (а если бы мы рассматривали 4 как self node, то 1 для 4 это parent) Т.е. тег в который вложен parent self node-a называется ancestor. Другими словами parent parent-a это ancestor.

5 - **preceding sibling** (предшествующий брат), этот элемент находится сверху в DOM по отношению к self node.

11 - **following sibling** (последующий брат), этот элемент находится внизу в DOM по отношению к self node. 4, 5, 11, все являются вложенными в один общий элемент № 4 (parent) и стоят на одной ступени иерархии

7, 9, 10 - **child**-s элементы по тношению к 6, т.е. 6 это parent для 7, 9, 10.

8 - **descendant** (потомок) для 6. Или 7,9,10 это тоже descendant для 4. А 6 descendant для 1. Другими словами child of child is descendant. Или ещё другими словами grandchild (правнук) of self node is descendant.

2 is a child for 1

3 is descendant for 1

2 and 3 are no related to **self node**

**Locating parent element**

**Синтаксис**:

**//известный\_нам\_иксПасс/parent :: название\_перента**

**//известный\_нам\_иксПасс/parent :: \*** ←если не знаешь название parent тега

**//известный\_нам\_иксПасс/ancesstor :: \*** ← так найдёт ансестора, но ансесторов будет несколько

**//известный\_нам\_иксПасс/.. ←** это сокращение, этот синтаксис тоже найдёт **перента**

Далее пример из DOM:

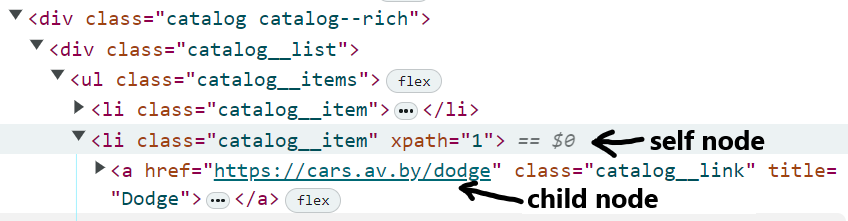


1)-нам надо найти элемент ‘a’, представим что у него нет атрибутов, мы можем взять его child элемент <span>,

тогда икс пасс будет:

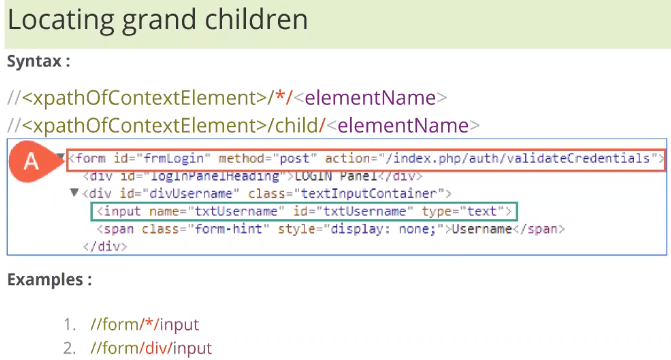
**//span[contains(text(),'Dodge')]/parent::a**

**Locating child element**



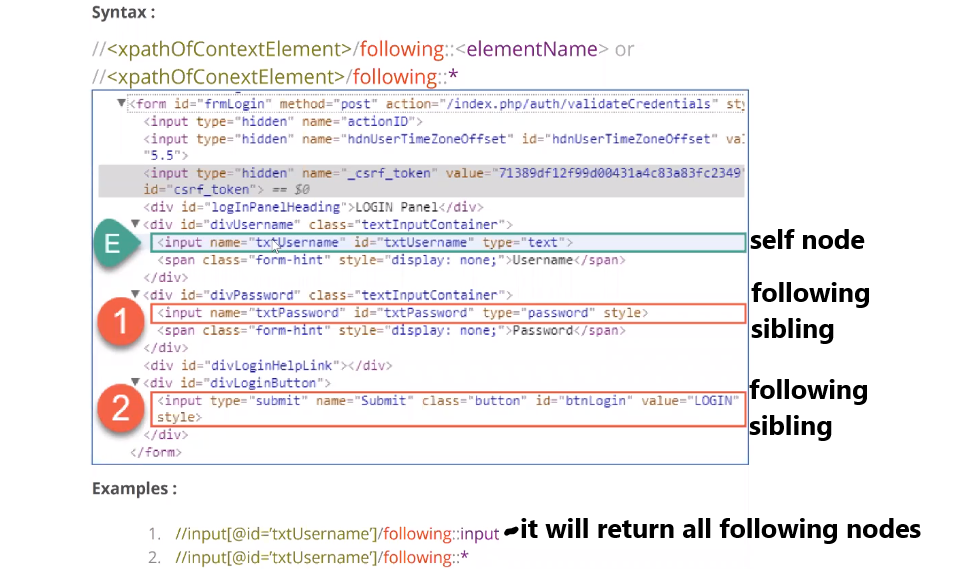
**//li[@class='catalog\_\_item']/child::a**

**//li[@class='catalog\_\_item']/a**  ← тоже самое но сокращённо, без (child::)

 ← other words descendent

с ancestor - тоже самое, такой же синтекс, смотри parent

**Locating following / preceding element**



if you want to return only specific element, use squire brackets with a number [4].

**//input[@id = ‘txtUserName’]/following::input[4]**

**//input[@id = ‘txtUserName’]/following::\*** it will return all following nodes.

For preceding elements sintex is the same

**//input[@id = ‘txtUserName’]/preceding::input**

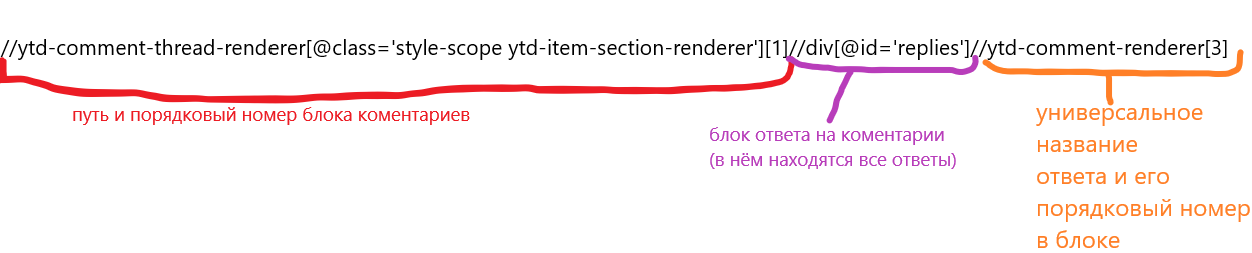
**//input[@id = ‘txtUserName’]/preceding::\***

------------------------------------------------------- **Поиск нескольких элементов** ----------------------------------------------

Ловил ошибку при нахождении, так как если ищем вложенный элемент/элементы, то записываем путь до этого данного элемента, но самый верхний элемент **div,** находится по его атрибуту **class**, именно значение этого атрибута берётся в квадратные скобки, а далее просто через слэш.

List <WebElement> elements=driver.findElements(By.xpath("//**div[@class='data-container']/ul/li**"));

—----------------------------------------**После написания botFindera**--------------------------------------------------------



На рисунке выше показан путь построения до уникального элемента, очень важно отметить что пользуясь инструментом разработчика (f12) при нахождении элемента нужно иметь ввиду, что если ты кликнул по элементу и получил его тэг с атрибутами и значениями, то это не факт что данный элемент один на странице и возможно проставив порядковый номер в конце xPath-а ничего не даст. В этом случае нужно вручную возвращатся к более старшему элементу который также будет иметь несколько копий, что надо учитывать пользуясь порядковым номером. **В целом помни, что при ненахождении нужного элемента (например их много) надо откатить к более старшим и крупным и возможно на их уровне и выстроить порядковый номер, что приведёт тебя к желаемому элементу дочернему**

—-------------------------------------------------------------------------------------------------------------------------------------------------------------

Ссылка на родительский узел (..) используется для ссылки на родительский узел узла контекста.

//div/ul[@id="longNumber"]/li[1] данный XPath дойдет до первого li[1]

//div/ul[@id="longNumber"]/li[1]/.. вернется на один уровень выше. Этот XPath равен XPath //div/ul[@id="longNumber"]

//div/ul[@id="longNumber"]/li[2] зайдет во второй li[2]

-----------------------------------------------------------------------------------------------------------

Группировка операций '()'

Группирует операции для явного задания порядка их выполнения.

Например: //div[4] вернет все (<div>) которые находятся на 4-ой позиции

(//div)[4] ввернет 4-ый элемент из коллекции элементов, т.е. выражение (//div) создаст коллекцию из всех элементов (<div>).

-----------------------------------------------------------------------------------------------------------

ancestor:: — возвращает множество предков.

//ul[@id="number"]/li/ancestor::\* вернет всех предков тега (<li>)

-----------------------------------------------------------------------------------------------------------

ancestor-or-self:: — возвращает множество предков и текущий элемент.

//ul[@id="number"]/li/ancestor-or-self::\* вернет всех предков тега (<li>) и сам тег (<li>)

-----------------------------------------------------------------------------------------------------------

attribute:: можно заменить на '@' — возвращает множество атрибутов текущего элемента,

//attribute::class вернет все элементы в которых содержится атрибут class

----------------------------------------------------------------------------------------------------------

Use Multiple attributes to locate an element.

Xpath- //button[starts-with(@id, 'save') and contains(@class,'publish')]

—---------------------------------------------------------------------------------------------------------------------------------------------------------------